REMARKS/ARGUMENTS

Claims 1-20 are pending in the present application. By this response, claims 1, 4-11, and 14-20 were amended. Support for the amendment to claims 1, 4-11, and 14-20 can be found in the claims as originally written. In addition, support for the amendment to claims 1, 11, and 20 can be found in the specification at least at page 5, line 18 through page 6, line 8. Reconsideration of the claims is respectfully requested.

I. Double Patenting Rejection

In the Final Office Action, the Examiner has provisionally rejected claims 5, 6, 15, and 16 on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claim 1 of copending Application No. 10,777,743 (hereinafter '743). More specifically, the Examiner stated:

> **As per instant claim 5**, '743 claim 1 also recites first call tree for first build and second call tree for second build, copying a first tree data structure; subtracting the call tree for the second build from the call tree for the first build; outputting the subtracted call tree, walking the second build tree structure over the first build tree structure to generate a subtracted tree; wherein for each node that exists in both the copied call tree for the first build and the call tree for the second build, generating a node in the subtracted tree data structure by subtracting a base value of the node in the second build tree from a base value of a corresponding node in the copied first call tree. Although '743 claim 1 does not recite exact wording (e.g. first build minimized call tree structure versus first call tree structure) as instant claim 5, the difference between the respective teaching regarding for example, second call tree and second build call tree structure would have been a obvious limitation. '743 discloses traversal with specific intent to computer difference among nodes visited; but does not recite inserting a pass field in each node of the copied first tree structure and initializing such node; but based on the walking algorithm as entailed from '743, well-known practices for initializing of a node every time a node (e.g. with a flag or indicative value) is visited (in order to compute a value during this traversal as in '743) would have been a obvious implementation in '743 by which algorithmic iteration step would be prevented from visiting a node more than needed based on such indicator than varies with the iteration.
>
> **As per instant claim 6**, '743 claim 1 also recites first call tree for first build and second call tree for second build, copying a first tree data structure; subtracting the call tree for the second build from the call tree for the first build; outputting the subtracted call tree, wherein for each node that exists only in the second build call tree structure, generating a node in the subtracted tree data structure having a negative value corresponding to a base value of the node that exists only in the second build tree structure, including the rationale of obviousness regarding insertion and initializing of pass field limitations. Although '743 claim 1 does not recite exact wording as instant claim 6, the difference between the respective teaching regarding first build copied call tree and second build second tree structure would have been a obvious limitation.

> **As per instant claim 15**, this is computer medium version of instant claim 5, reciting the same limitations therein, hence instant claim 15 would have been a obvious variation of'743 claim 1, by virtue of the above analysis.
>
> **As per instant claim 16**, this is computer medium version of instant claim 6, reciting the same limitations therein, hence instant claim 16 would have been a obvious variation of'743 claim 1, by virtue of the above analysis.

Final Office Action dated April 10, 2008, pp. 3-4 (emphasis in original).

In response, Applicants believe that the submission of a terminal disclaimer may be premature given the fact that the Examiner has made no indication as to whether claims of '743 application are allowable. If at a later time, such an indication is made, Applicants may then elect to provide a terminal disclaimer for overcoming this provisional double patenting rejection.

II. 35 U.S.C. § 112, First Paragraph

In the Final Office Action, the Examiner rejected claims 1-20 for failing to comply with the written description requirement. This rejection is respectfully traversed. More specifically, the Examiner stated:

> As recited in claims 1, 11, 20 the scenario of having a first call tree then a copied call tree structure and using this copied version to generate a subtracted call tree is deemed a lack of teaching or insufficient support from the Disclosure.
>
> Claim 1 recites 'obtaining ~ first call tree' then 'copying the first call tree to form ~ copied call tree data structure' then 'generate ~ subtracted call tree structure', leading thereby to the understanding that a copied first call tree is separate entity from the first call tree, that a subtracted tree is a distinct tree resulting from the subtracting process using both the second tree and the above copy (of the first tree); that is, at least 3 trees (not counting the second call tree) are necessarily part of the claimed invention: a first call tree, a copy thereof, a subtracted tree. However, based on the corresponding support from the Specification, what is interpreted as a concurrent existence of (i) first call tree, (ii) 'copied version of a first call tree' and (iii) 'subtracted call tree' is deemed not credible a teaching as to enable one of ordinary skill in the art to make use of the above teaching, or to acknowledge that the inventor does possess all the entities (i) (ii) and (iii) based on the respective order of appearance thereof in the claim.
>
> According to the Specifications a process starts with creating a copy of a first call tree as target build A (Specifications: *generating a copy of the call tree for the first build A* - bottom half, pg. 5 to first half pg. 6; bottom pg. 34), then based on the call tree of the build B (second call tree structure), walking to compare Build A tree to Build B tree is performed. Hence, the generating of build A tree would be interpreted as a first call tree being obtained. Absent any slightest detail about how this build A tree would be used to create a duplicate tree, this description does not sufficiently teach one of ordinary skill in the art that a copying process is performed so that all the content of this build A tree thus created is duplicated to generate another build A 'copied call tree'. Moreover, any node whose value is being set based on comparing between respective nodes in the build A tree and in build B tree happens to belong to the

build A tree (first call tree), during the walking process *(value for this node is set* - bottom pg. 34; Fig. 9B); and, as disclosed, the resulting tree wherein all nodes have been set to a value would be the result of, this very build A tree. That is, there no mention of creation of a copy for build A tree, nor is there creation of a separate third tree structure (emphasis added) being actually disclosed when this node is set to a base value -- or a node is created exactly at very location where the 'copied tree' (i.e. the first call tree) is being compared (e.g. Specifications: bottom pg. 34, top pg. 35). As disclosed in the walking down wherein only two call tree structures, a Build A tree and a Build B tree, are involved (with the former being modified to yield a subtracted end resulting tree) the entities as claimed -- (i), (ii), and (iii) - are deemed not in possession by the Inventor at the time the Invention was made. Thus, the *copied tree* recited as (ii) and the obtaining of (i) would be treated as one tree from generating said first call tree, and the generating of (iii) would be treated as mere modifying of an existing tree (e.g. the first call tree) based on the comparison process between the second tree and the first tree.

Claims 11, and 20 exhibit language that entails existence of tree structures (i) and (ii) along with the subtracted tree structure (iii); hence would also be rejected for lack of enabling description based on the above analysis. The entities (ii) and (iii) would be treated as slight language variant representing to the same first call tree (i) being initially created or obtained.

Final Office Action dated April 10, 2008, pp. 4-5 (emphasis in original).

In response, claims 1, 11, and 20 have been amended to remove references to a copied call tree data structure. Therefore, the rejection of claims 1, 11, and 20 under 35 U.S.C. § 112, first paragraph has been overcome.

III. 35 U.S.C. § 103, Obviousness

The Examiner has rejected claims 1-20 under 35 U.S.C. § 103(a) as being unpatentable over *Levine et al.*, U.S. Patent No. 6,349,406 (hereinafter "*Levine*") in view of *Reissman et al.*, U.S. Publication No. 2005/00171818 (hereinafter "*Reissman*"). This rejection is respectfully traversed.

In rejecting claim 1 the Examiner states:

As per claim 1, Levine discloses a method, in a data processing system, for identifying differences between the execution of a first computer program and a second computer program, comprising:

obtaining a first call tree data structure corresponding to first trace data of an execution of the first computer program (e.g. first pass - Fig. 21; *call stack tree* representation - col. 23, lines 1-8; Fig. 22);

obtaining a second call tree data structure corresponding to second trace data of an execution of the second computer program (e.g. second pass - Fig. 21 - Note: walking a stack tree representing events with entry and exit of functions **called** based on a trace record reads on a call tree structure - see col. 10, lines 15-22; col. 12, lines 15-47 - and a walking based on a different trace reads on a second call tree);

copying the first call tree data structure to form a copied call tree data structure (NOTE: the generating of a first structure tree by Levine to enable to

comparison step to start would read on having created a copy of a tree to consider for walking against the second tree structure --**see interpretation as set forth in the USC § 112, 1st para Rejection**, where first tree, copy of first tree and the resulting tree - subtracted tree - are subsumed in one tree created as one initial first tree being used for the walking process)

subtracting the second call tree data structure from the first call tree data structure to generate a subtracted call tree data structure (delta event - Fig. 20A; if there are more traces - co1.18, lines 7-25 - Note: traces to process reads on first trace and second trace with corresponding call tree each); and

outputting the subtracted call tree data structure, wherein the subtracted call tree data structure identifies differences (step 2312 - Fig. 23A; col. 22, line 25-55 - Note: delta time node associated to node of stack tree reads on outputting delta represented by stack tree structure, hence created a *subtracted stack tree* - see step 2314, Fig. 23A) between the execution of the first computer program and the execution of the second computer program.

Levine does not explicitly disclose that the first computer program is a first build thereof and that the second computer program is second build thereof. Levine mentions about using differences that may happen for different platforms, e.g. as Java APIs -- so that trace code has to be used to effect profiling and reduction of code (col. 7, line 61 to col. 8, line 17; col. 9, line 30- 48) as endeavored by developers for enhancing code for a given hardware platforms (see BACKGROUND: col. 2, lines 50-58), hence has suggested code improvement made to accommodate software build for platforms. Reissman also discloses a tool to test APIs based on analysis of software builds via a scanning tool whereby dependencies tree is supporting recording of differences between method calls including execution state of these calls, based on such record or dictionary structure so to yield differences (see para 0041, pg. 4; Fig. 3). It would have been obvious for one skill in the art at the time the invention was made to implement the trace analysis and difference structure generating approach by Levine, so that the computer program being traced and converted into tree of events for profile are computer intended for builds as taught by Reissman, because executables intended for different environments (see Reissman) via improved builds can be addressed by profiling and analysis based on events as set forth by either Levine and Reissman, whereby based on such differences between trace profiling data, one build as well as testing techniques therefor can be readjusted for improvement over the previous build (see Reissman, BACKGROUND).

Final Office Action dated April 10, 2008, pp. 8-10 (emphasis in original).

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on prior art when rejecting claims under 35 U.S.C. § 103(a). *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). The scope and content of the prior art are determined; differences between the prior art and the claims at issue are ascertained; and the level of ordinary skill in the pertinent art resolved. *Graham v. John Deere Co.*, 383 U.S. 1 (1966). Against this background, the obviousness or non-obviousness of the subject matter is determined. *Id.* Often, it will be necessary for a court to look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person having ordinary skill in

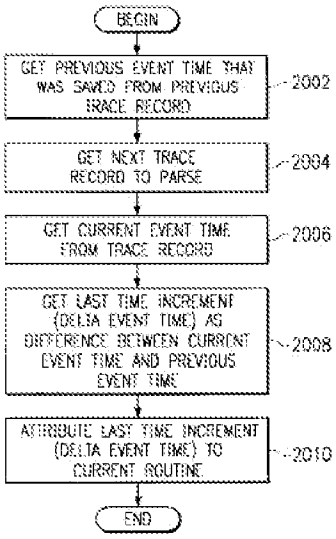
the art, all in order to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue. *KSR Int'l. Co. v. Teleflex, Inc.*, No. 04-1350 (U.S. Apr. 30, 2007). Rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness. *Id.* (citing *In re Kahn*, 441 F.3d 977, 988 (CA Fed. 2006)).

Amended claim 1 is as follows:

> 1. A computer implemented method for identifying differences between an execution of a first built computer program and a second built computer program, comprising:
>
> obtaining a first call tree data structure corresponding to first trace data of an execution of the first built computer program;
>
> obtaining a second call tree data structure corresponding to second trace data of another execution of the second built computer program;
>
> walking the second call tree data structure over the first call tree data structure to generate a third call tree data structure, wherein the third call tree data structure includes all nodes of both the first call tree data structure and the second call tree data structure, and wherein each node of the third call tree data structure includes a pass field having one of a first pass field value indicating that a first node was only present in the first call tree data structure, a second pass field value indicating that a second node was only present in the second call tree data structure, and a third pass field value indicating that a third node was present in both the first call tree data structure and the second call tree data structure; and
>
> outputting the third call tree data structure, wherein the third call tree data structure identifies differences between the execution of the first built computer program and the another execution of the second built computer program.

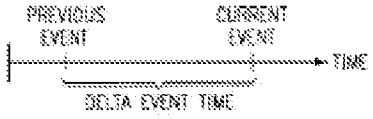
The Examiner has not stated a *prima facie* obviousness rejection of claim 1 because the proposed combination of references, when considered as a whole, does not teach or suggest the feature "walking the second call tree data structure over the first call tree data structure to generate a third call tree data structure, wherein the third call tree data structure includes all nodes of both the first call tree data structure and the second call tree data structure, and wherein each node of the third call tree data structure includes a pass field having one of a first pass field value indicating that a first node was only present in the first call tree data structure, a second pass field value indicating that a second node was only present in the second call tree data structure, and a third pass field value indicating that a third node was present in both the first call tree data structure and the second call tree data structure," as recited by amended claim 1.

Levine does not disclose generating the third call tree data structure despite the fact that *Levine* mentions a "delta event" in Figure 20A, which the Examiner cited to in asserting that *Levine* disclosed a subtracted call tree data structure, as claimed in original claim 1. Figure 20A is as follows:



Levine, Figure 20A.

According to *Levine*, "Figure 20A is a flowchart depicting a summary of the manner in which elapsed time is attributed to various routines in an execution flow." (*Levine*, col. 4, ll. 36-38.) In particular, *Levine* indicates that the delta event is calculated as a difference between a current event time and a previous event time. Importantly, *Levine* indicates that the current event and the previous event time are from a single trace file. For example, *Levine* discloses that Figure 20A is a process for "determining how to interpret timestamps written to **a trace file** and attribute the time flow recorded in **a trace file**." (*Levine*, col. 22, ll. 22-24, emphasis added.) Figure 20B, which provides additional insight into the determination of the delta event time, is as follows:



Levine, Figure 20B.

Figure 20B depicts the delta time event referenced in Figure 20A. Regarding Figure 20B, *Levine* provides:

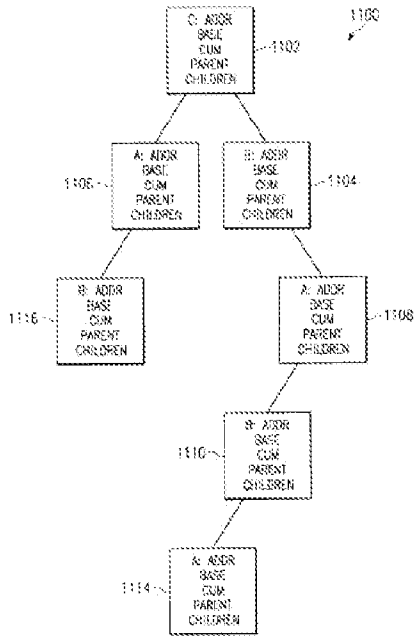
> The previous event time is the timepoint at which a previous event has occurred, such as an entry event or an exit event. The current event is the timepoint at which the subsequent event has been recorded to occur, such as the next entry event or exit event.

Levine, col. 22, ll. 42-46.

Thus, the delta event time is a value calculated from timestamps of **a single trace file**, rather than by walking a second call tree data structure over a first call tree data structure to generate a third call tree data structure, as recited by amended claim 1. Importantly, claim 1 recites that the second call tree data structure corresponds to second trace data of another execution of the second built computer program.

Thus, because *Levine* discloses a delta event from only a single trace file, the delta event of *Levine* does not teach or suggest the walking feature of claim 1 when read in light of claim 1 as a whole. Consequently, *Levine* does not teach or suggest this missing "walking" feature recited by amended claim 1.

In addition, *Levine* does not disclose a third call tree data structure including, in each node, a pass field having one of a first pass field value indicating that the a first node was only present in the first call tree data structure, a second pass field value indicating that a second node was only present in the second call tree data structure, and a third pass field value indicating that a third node was present in both the first call tree data structure and the second call tree data structure, as recited by amended claim 1. For example, *Levine* includes only two call tree data structures, neither of which includes a pass field as recited in claim 1. The first tree data structure of *Levine* is as follows:

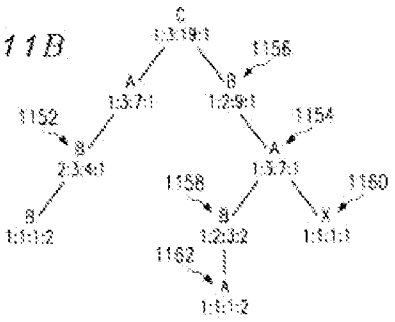


Levine, Figure 11A.

According to *Levine*, Figure 11A is a "tree structure generated from sampling a call stack." (*Levine*, col. 4, ll. 10-11.) The corresponding text of Figure 11 indicates that the nodes contain an address, a base, a cumulative time, and parent and children pointers. Nowhere in Figure 11A nor in the accompanying text does *Levine* describe a pass field having one of a first pass field value indicating that a first node was only present in the first call tree data structure, a second pass field value indicating that a second node was only present in the second call tree data structure, and a third pass field value indicating that a third node was present in both the first call tree data structure and the second call tree data structure, as recited by amended claim 1. Similarly, the second tree data structure is also devoid of mention to a pass field as claimed in amended claim 1.

The second tree data structure of *Levine* is as follows:

FIG. 11B



Levine, Figure 11B.

Figure 11B is a "diagram depicting an event tree which reflects call stacks observed during system execution." (*Levine*, col. 4, ll. 12-13.) According to *Levine*, the type of information maintained in the event tree of Figure 11B may include "time based statistics," (*Levine*, col. 16, ll. 33-42) "bytecodes executed," (*Levine*, col. 17, ll. 24-25) and an "amount of memory allocated." (*Levine*, col. 17, ll. 31-32.) However, *Levine* makes no reference to a pass field as claimed in amended claim 1. Therefore, because no portion of *Levine* discloses the pass field as claimed in amended claim 1, *Levine* does not teach or suggest the feature, "walking the second call tree data structure over the first call tree data structure to generate a third call tree data structure, wherein the third call tree data structure includes all nodes of both the first call tree data structure and the second call tree data structure, and wherein each node of the third call tree data structure includes a pass field having one of a first pass field value indicating that a first node was only present in the first call tree data structure, a second pass field value indicating that a second node was only present in the second call tree data structure, and a third pass field value indicating that a third node was present in both the first call tree data structure and the second call tree data structure," as recited by amended claim 1.

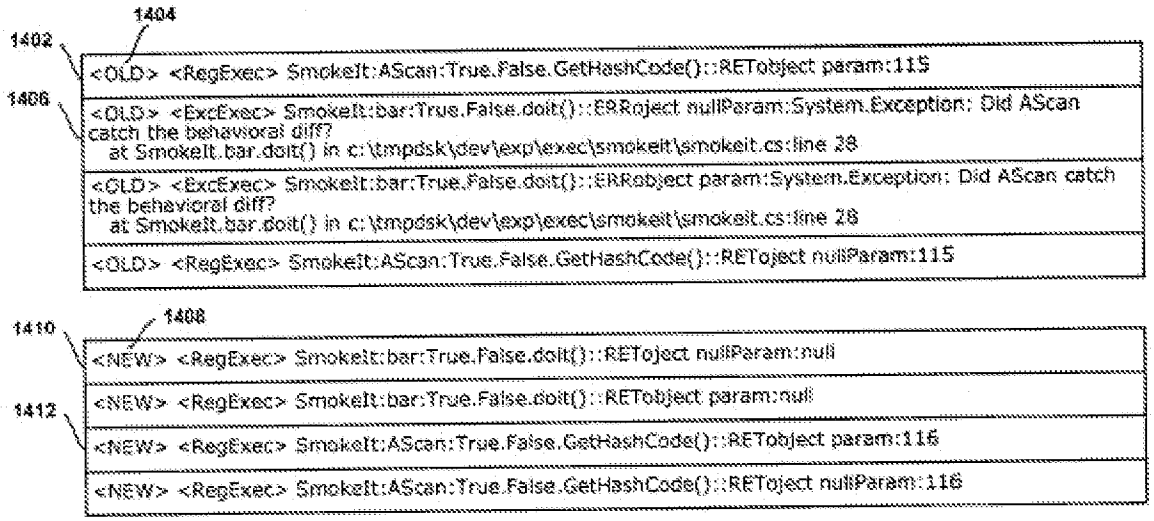
Because *Reissman* does not cure the deficiency of *Levine*, the proposed combination of references, when considered as a whole, does not teach or suggest all the features of amended independent claim 1. Like *Levine*, *Reissman* does disclose the "walking" feature of amended claim 1. *Reissman* is directed to a system and method for testing software builds by comparing dictionaries. To this end, *Reissman* discloses recording differences between two dictionaries. For example, *Reissman* discloses:

> FIG. 14 presents an illustration generally representing an example of a delta computed between two software builds. The differences discovered by a comparison between a new dictionary and an old dictionary may be recorded by listing the dictionary entry with the prefix <OLD> if deleted from the new dictionary and listing the dictionary entry with the prefix <NEW> if added to the new dictionary as shown in FIG. 14. For example, dictionary entry 1402 has the prefix <OLD> 1404 and indicates an entry that has been deleted from the new

dictionary. Likewise, dictionary entry 1412 has the prefix <NEW> and indicates an entry that has been added to the new dictionary.

Reissman, p. 7, para. 66.

An example of the recorded differences is as follows:



Reissman, Figure 14.

Figure 14 is a diagram of differences recorded from a comparison of two dictionaries for different software builds. Importantly, *Reissman* discloses that only **the differences between** the dictionaries are recorded. Amended claim 1, on the other hand, recites that the third call tree data structure includes **all nodes** of both the first call tree data structure and the second call tree data structure. Thus, because *Reissman* only discloses recording of the differences between the two dictionaries, *Reissman* does not teach or suggest a third call tree data structure having all the nodes of the first and second dictionaries. Moreover, *Reissman* does not disclose outputting the differences as a call tree data structure. For these reasons, the proposed combination of references, when considered as a whole, does not teach or suggest the feature, "walking the second call tree data structure over the first call tree data structure to generate a third call tree data structure, wherein the third call tree data structure includes all nodes of both the first call tree data structure and the second call tree data structure, and wherein each node of the third call tree data structure includes a pass field having one of a first pass field value indicating that a first node was only present in the first call tree data structure, a second pass field value indicating that a second node was only present in the second call tree data structure, and a third pass field value indicating that a third node was present in both the first call tree data structure and the second call tree data structure," as recited by amended claim 1. Because independent claims 11 and 20 also recite substantially the same feature as that of amended claim 1, the Examiner has also not stated a *prima facie* obviousness rejection as to claims 11 and 20 at least for the reasons set forth above. In addition, the Examiner has not stated a *prima facie*

obviousness rejection as to dependent claims 2-10 and 12-19 by virtue of their dependency from claims 1 and 11. Therefore, the rejection of claims 1-20 under 35 U.S.C. § 103(a) has been overcome.

IV. Conclusion

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: September 10, 2008

Respectfully submitted,

/Stephen Liu/

Stephen Liu
Reg. No. 62,883
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants